A Distributed Algorithm for Finding All Best Swap Edges Of a Minimum Diameter Spanning Tree*

DIPAK GOVARDHAN BHUSARI[1], PRASAD RAMKRISHNA KULKARNI[2]

[1] *Assistant Professor, CSE department HOD, Aditya Engineering College, Beed, Maharashtra.*
[2] *Assistant Professor, CSE department HOD, Aditya Polytechnic College Beed,, Maharashtra.*

ABSTRACT

Communication in networks suffers if a link fails. When the links are edge of a tree that has been chosen from an underlying graph of all possible links, a broken link even disconnect the network. Most often, the link is restored quickly. A good policy to deal with this sort of transient link failures is swap rerouting s, where the temporarily broken link is replaced by a single swap link from the underlying graphs. A rapid replacement of a broken link by a swap link is only possible if all swap links have been precomputed. Then the selection of high quality swap links is essential; it must follow the same objective as the originally chosen communication subnetwork.

We are interested in a minimum diameter tree in a graph with edge weights (so as to minimize the maximum travel time of messages). Hence, each swap link must minimize (among all possible swaps) the diameter of the tree that results from swapping. We advice a distributed algorithm that efficiently computes all of these swap links and we explain, how to route message across swap edges with a compact routing scheme. lastly, we consider the computation of swap edges in an arbitrary spanning tree, where swap edges are chosen to reduce the time required to adapt routing in case of a failures, and give efficient distributed algorithms for two variants of this problem.

.

1. INTRODUCTION

For communication in computer networks, often only subsets of the available connections are used to communicate at any given time. If all nodes are associated using the smallest number of links, the subset form a spanning tree of the network. This has economical benefits compare to using the entire set of available links, assuming that the merely keeping a link active for potentially sending messages induces some cost. Additionally, as only one path exists between any communication pair, a spanning tree simplifies the routing and allows small routing tables.

Depending on the purpose of the networks, there is a variety of desirable properties of spanning trees. We are interested in a Minimum Diameter Spanning Tree (MDST), i.e., a tree that minimizes the largest distance between any pair of nodes, hence minimizing the worst case length of any transmission paths, even if edge lengths are not uniforms. The importance of minimizing the diameter of a spanning tree has been widely recognized, and then the diameter of a network provides a lower bound (and often even an exact one) on the computation time of most algorithms in which all nodes participates. One downside of using a spanning tree is that a single link failure disconnects the network.

Whenever the link failures are transient, i.e., a failed link shortly becomes operational another time, the momentarily best possible way of reconnecting the network is to replace the failed link by a single other link, it is called a swap link. Among all possible swap links, one should be choose a best swap w.r.t. the original intention that is in our case, a swap that minimize the diameter of the resulting swap trees.

2. RELATED WORK

Centralized algorithm for computing all best swaps of a MDST in O $O(n\sqrt{m})$ time and O(m) space, the given underlying communication network G = (V,E) has n = |V | vertices and m = |E| edges. For shortest path trees, an earlier centralized algorithm has been complemented by a distributed algorithm using totally different techniques for finding all best swap edges for several objectives, with either O(n) messages of size O(n) each, or O(n*) short messages with size O(1) each, where n* denotes the size of the transitive closure of the tree, where the edges are directed away from the roots. In a so-called preprocessing phase of this algorithm, several information is computed along with the spanning tree construction using O(m) message.

A distributed algorithm for computing a MDST in a graph G(V,E) in an asynchronous setting has O(n) time complexity (in the standard sense, as explained in Section 3) and uses O(nm) messages. However, no efficient distributed algorithm to compute the best swaps of a MDST had been found to date.

In this paper, we propose a distributed algorithm for computing all best swaps of a MDST using no more than O(max{n*,m}) messages of size O(1) each. Size of a message denotes the number of atomic values that it contains, such as edge weights, node labels, path lengths etc., and n* is size of the transitive closure of the MDST with edges directed away from a center of the tree. Both m and n* are very natural bounds: When each subtree triggers as many messages as there are nodes in the subtree, size of the transitive closure describes the total number of messages. Moreover, it seems inevitable that each node receives some information from each of its neighbors in G, across each potential swap edge.

Our algorithm runs in O(‖D‖)time (in the standard sense, as explained in Section 3), where ‖D‖ is the hop-length of the diameter path of G; note that this is asymptotically optimal. The message and time costs of our algorithm are easily subsumed by the costs of constructing a MDST distributively using the algorithm. Thus, it is cheap to precompute all the best swaps in addition to constructing a MDST initially. Just like the best swaps algorithms for shortest paths trees [3, 4], our algorithm (like many fundamental distributed algorithms) exploits the structure of the tree. This tree is substantially different in that it requires a significantly more complex invariant to be maintained during the computation: We need to have just the right collection of pieces of paths available so that on the one hand, then these pieces can be maintained efficiently, and on the other hand, they can be composed to reveal the diameter at the corresponding steps in the computation.

Furthermore, we propose a compact routing scheme for trees which can quickly and inexpensively adapt routing when a failing edge is replaced by a best swap edge. particularly, our scheme does not require an additional full backup table, but assigns a label of c log n bits to each node (for some small constant c); a node of degree δ stores the labels of all its neighbours (and itself), which amounts to mc log n bits or δc log n bits per node in total. Given this labelling, knowledge of the labels of both adjacent nodes of a failing edge and the labels of both adjacent nodes of its swap edge is sufficient to adjust routing

3. ALGORITHMIC SETTING AND BASIC IDEA

In our setting, nodes have unique identifiers that possess a linear order. More, let each node know its own neighbours in T and in G, and for each neighbor the length of the corresponding edge. We assume port-to-port communications between neighbouring nodes. Then the distributed system of nodes is totally asynchronous. Each message sent from some node to one of its neighbours eventually arrives (there is no message loss). As usual, we defines the asynchronous time complexity of an algorithm as the longest possible execution time assuming that sending a message requires at most onetime unit. Furthermore, nodes do not need to know the total number of nodes in the system (although it is easy to count the nodes in T using a convergecast).

3.1 The Basic Idea

Our goal is to compute, for each edge of T as a best swap edge. A swap edge for a given failing edge e = (x, p(x)) must connect the subtree of T rooted at x to the part of the tree containing p(x). So, a swap edge must be adjacent to some node inside T_x. If each node in T_x consider its own local swap edges for e, then all swap edges for e are considered. For that reason, each node inside T_x finds a best local swap edge, and participates in a minimum finding process that computes a (globally) best swap edge for e. then in total all swap edges for e are considered. Therefore, each node inside T_x finds a best local swap edge, and it participates in a minimum finding process that computes a (globally) best swap edge for e. The computation of the best local swap edges is composed of three main phases: In a first preprocessing phase, the root of the MDST is chosen, and various pieces of information (explained later) are computed for each node. In a top-down phase each node computes and forwards some “enabling information” (explained later) for each node in its own subtree. This information is merged and collected in a third bottom-up phase, during which each node obtain its best local swap edge for each (potentially failing) edge on its path to the root. Then the efficiency of our algorithm will be due to our careful choice of the various pieces of information that we collect and use in these phases.

To give an overview, we now briefly sketch how each node computes a best local swap edge. First observes that after replacing edge e by f, a resulting diameter is longer than the previous diameter only if there is a path through f which is longer than the previous diameter, in that case the path through f is the new diameter. In this case, length of the diameter equals the length of a longest path through f in the new tree. For a local swap edge f = (z, z') connecting node $z \in V(T_x)$ and $z' \in V \setminus V(T_x)$, such a path consists of

(i) a longest path inside T\ T_x starting in z'.

(ii) edge f.

(iii) a longest path inside T_x starting in z.

Part (i) is computed in a preprocessing phase, as described in Section 5. Part (ii)is by assumption known to z, because f is adjacent to z. Part (iii) is inductively computed by a process starting from the root x of Tx, and stopping in the leaves, as follows. A path starting in z and staying inside Tx either descends to a child of z (if any), or goes up to p(z) (if p(z) is still in Tx) and continues within Tx $\backslash T_z$.

For the special case where z = x, node x needs to consider only the heights of the subtrees rooted at its children. All other nodes z in T_x additionally needs to know the length of a longest path starting at p(z) and staying inside $T_x \backslash T_z$. Then this additional enabling information will be computed by p(z) and then be sent to z. Once the best local swap edges are recognised, a best (global) swap edge is identified by a single minimum finding process that starts at the leaves of T_x and ends in node x. To calculate all best swap edges of T, then this procedure is executed separately for each edge of T.

4. ROUTING ISSUES

A natural question arises concerning routing in the presence of a failure: After replacing the failing edge e by a best swap edge f, how do we adjust our routing mechanism in order to guide messages to their destination in the new tree T e/f ? And how is routing changed back again after the failing edge has been repaired? Clearly, it is desirable that the adaptation of the routing mechanism is as fast and inexpensive as possible.

Our routing scheme for trees is based on the labeling $\gamma : V \rightarrow \{1, \ldots, n\}^2$ described in the end of Section 5. Note that γ allows to decide in constant time whether a is in the subtree of b (i.e., $a \in T_b$) for any two given nodes a and b.

Routing Algorithm:

A node s routes message M with destination d as follows:

(i) If d = s, M has arrived at its destination. (ii) If $d \,/\!\in T_s$, s sends M to p(s).

(iii) Otherwise, s sends M to the child $q \in C(s)$ for which $d \in Tq$.

This algorithm clearly routes each message directly on its (unique) path in T from s to d. Before describing the adaptation in the presence of a swap, observe that a node s which receives a message M with destination d can locally decide whether M traverses a given edge e = (x, p(x)): edge e is used by M if and only if exactly one of s and d is in the subtree Tx of x, i.e., if $(s \in Tx) \neq (d \in Tx)$.

Thus, it is enough to adapt routing if all nodes are informed about a failing edge (and later the repair) by two broadcasts starting at its two incident nodes (the points of failure). However, the following lemma shows that optimal rerouting is guaranteed even if only those nodes which lie on the two paths between the points of failure and the swap edge's endpoints are informed about failures, which allows "piggybacking" all information for routing adjustment on the first message arriving at the point of failure after the failure occurred.

5. ACKNOWLEDGEMENT

Our thanks to the experts who have contributed towards the development of this paper.

6. REFERENCES

[1]. M. Andrews, B. Awerbuch, A. Fern´andez, T. Leighton, Z. Liu, and J. Kleinberg. Universal-stability results and performance bounds for greedy contentionresolution protocols. J. ACM, 48(1):39–69, 2001.

[2]. M. Bui, F. Butelle, and C. Lavault. A Distributed Algorithm for Constructing a Minimum Diameter Spanning Tree. Journal of Parallel and Distributed Computing,

64:571–577, 2004.

[3]. P. Flocchini, A. M. Enriques, L. Pagli, G. Prencipe, and N. Santoro. Point-of-failure Shortest-path Rerouting: Computing the Optimal Swap Edges Distributively. IEICE Transactions on Information and Systems, E89-D(2):700–708, 2006.

[4]. P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, and P. Widmayer. Computing All the Best Swap Edges Distributively. In Proceedings of Symposium on Principles of Distributed Systems (OPODIS), volume 3544 of Lecture Notes in Computer Science, pages 154–168, 2004.

[5]. H. Ito, K. Iwama, Y. Okabe, and T. Yoshihiro. Single Backup Table Schemes for Shortest-path Routing. Theoretical Computer Science, 333(3):347–353, 2005.

[6]. E. Nardelli, G. Proietti, and P. Widmayer. Finding All the Best Swaps of a Minimum Diameter Spanning Tree Under Transient Edge Failures. Journal of Graph Algorithms and Applications, 5(5):39–57, 2001.

[7]. E. Nardelli, G. Proietti, and P. Widmayer. Swapping a Failing Edge of a Single Source Shortest Paths Tree Is Good and Fast. Algorithmica, 35(1):56–74, 2003.

[8]. A. Di Salvo and G. Proietti. Swapping a Failing Edge of a Shortest Paths Tree by Minimizing the Average Stretch Factor. In Proceedings of 11th Colloquium on Structural Information and Communication Complexity (SIROCCO), 2004.

[9]. N. Santoro. Design and Analysis of Distributed Algorithms. Wiley Series on Parallel and Distributed Computing. Wiley, 2007.

[10]. N. Santoro and R. Khatib. Labelling and Implicit Routing in Networks. The Computer Journal, 28(1):5–8, 1985.

BIOGRAPHIES

	Mr.DIPAK GOVARDHAN BHUSARI was completed B.E (CSE) degree from Babasaheb Ambedkar Marathwada University (BAMU), Aurangabad and M.Tech degree from sana engineering college,kodad,nalgonda Dist.(Hyderabad). Working as HOD(CSE) at Aditya ENGINEERING college Beed. Maharashtra.
	Mr. Prasad Ramkrishna Kulkarni was completed B.E (CSE) degree from Babasaheb Ambedkar Marathwada University (BAMU), Aurangabad and M.Tech degree from sana engineering college,kodad,nalgonda Dist.(Hyderabad). Working as HOD(CSE) at Aditya polytechnic college Beed. Maharashtra.